A Proof with Side Effects of Gödel’s Completeness Theorem Suitable for Semantic Normalisation

Hugo Herbelin

PPS, INRIA and University 7, France
hugo.herbelin@inria.fr

As revealed by T. Griffin [3], classical reasoning is related to control operators. Classical reasoning can be simulated within intuitionistic logic via a negative translation, the same way programming with control can be simulated in pure $\lambda$-calculus by reasoning within a continuation monad. Under this view, classical logic is “direct style” for proving intuitionistically within the target of a negative translation.

We shall consider a monotonic memory update effect and connect it to Kripke’s forcing, seen as a dependent variant of the reader monad. We shall then interpret reasoning within the target of Kripke’s forcing as indirect style for reasoning with a monotonic memory update effect (by monotonic update is meant that the memory can be updated only with a value which refines the previous value according to a given refinement order).

As an application, let us consider logical completeness of classical logic with respect to two-valued models and logical completeness of intuitionistic logic with respect to Kripke models and see the former as a direct-style formulation of the latter using monotonic memory update.

The core of the completeness proof proves $(\models_{\mathcal{M}} A) \iff (\Gamma \vdash A)$ for the two-valued model defined on atoms by $\models_{\mathcal{M}} P \equiv (\Gamma \vdash P)$, where $\Gamma$ is a mutable variable denoting a context and updatable according to the context extension order.

It is formulated in an intuitionistic second-order arithmetic with delimited side-effects validating a specific principle characterising monotonic memory update. This arithmetic is intuitionistic in the sense that the disjunction and witness existence properties are retained.

The proof instructions for delimited monotonic update are set and update where $T$ ranges hereditarily positive formulas, i.e. over $\forall \rightarrow$-free formulas. The updatable variable is denoted by an ordinary term variable $x$.

\[
\begin{align*}
\Gamma, [b : x \geq t] \vdash q : T(x) & \quad \Gamma \vdash r : refl \geq x \text{ fresh in } \Gamma \text{ and } T(t) \\
\Gamma \vdash \text{set } x := t \text{ as } b/_{(r,s)} \text{ in } q : T(t) \quad & \text{SETEFF} \\
\Gamma, [b : x \geq t(x')] \vdash q : T(x) & \quad \Gamma \vdash r : t(x') \geq x' \quad [x \geq u] \in \Gamma \text{ for some } u \quad x' \text{ fresh in } \Gamma \\
\Gamma \vdash \text{update } x := t(x) \text{ of } x' \text{ as } b \text{ by } r \text{ in } q : T(t(x)) \quad & \text{UPDATE}
\end{align*}
\]

The core of the completeness proof is defined for the negative fragment of classical logic by mutual induction using the “reify” $\downarrow$ and “reflect” $\uparrow$ functions used in the context of semantic normalisation (starting with [1]) and type-directed partial evaluation (e.g. [2]).
The core of the completeness proof is:

\[
\begin{align*}
\uparrow_A & \quad \Gamma \vdash A \quad \rightarrow \quad \models_M A \\
\uparrow_{p(t)} & \quad g \quad \triangleq \quad g \\
\uparrow_{A \rightarrow B} & \quad g \quad \triangleq \quad m \mapsto \uparrow_B \text{App}_{\Gamma, A, B}(g, \downarrow_A m) \\
\uparrow_{\forall x \ A} & \quad g \quad \triangleq \quad t \mapsto \uparrow_{A[t/x]} \text{App}_{\Gamma, x, A}(g, t) \\
\downarrow_A & \quad \models_M A \quad \rightarrow \quad \Gamma \vdash A \\
\downarrow_{p(t)} & \quad m \quad \triangleq \quad m \\
\downarrow_{A \rightarrow B} & \quad m \quad \triangleq \quad \text{Abs}_{\Gamma, A}^{\Gamma_1, A, (\text{update } \Gamma := (\Gamma_1, A) \text{ of } \Gamma_1 \text{ as } b_A \text{ by } r_A \text{ in } \downarrow_B (m (\uparrow_A \hat{\text{Ax}}^{\Gamma_1, A, \Gamma}(b_A)))))} \\
\downarrow_{\forall x \ A} & \quad m \quad \triangleq \quad \text{Abs}_{\Gamma, x, A}^{\Gamma, \Gamma_1, A, (\hat{y}, \downarrow_{A[z/x]} (m \hat{y})))}
\end{align*}
\]

where \(\text{Abs}_\rightarrow\), \(\text{App}_\rightarrow\), \(\text{App}_\forall\) and \(\hat{\text{Ax}}\) are the constructors of the logic while \(r_X\) proves \(\Gamma \subset (\Gamma, X)\) and \(\hat{y}\) is taken fresh in \(\Gamma\).

Let \(\text{Classic}(\mathcal{M}) \triangleq \forall A (\models_M \neg \neg A \quad \rightarrow \quad \models_M A)\). The next step is to show that the model is classical:

\[
\text{classic} : \quad \text{Classic}(\mathcal{M}) \\
\text{classic} \triangleq \quad A \mapsto m \mapsto \uparrow_A (\hat{\text{Dn}}(\downarrow_{\neg \neg A} m))
\]

with \(\hat{\text{Dn}}\) standing for double negation elimination.

The final statement is then:

\[
\text{compl}_A : \quad \forall \mathcal{M} \forall \sigma (\text{Classic}(\mathcal{M}) \rightarrow \models_M A) \quad \rightarrow \quad \vdash A
\]

\[
\text{compl}_A \triangleq \quad \psi \mapsto \text{set } \Gamma := \emptyset \text{ as } b/\{(r, s)\} \text{ in } \downarrow_A (\psi \mathcal{M} \text{ id } \text{classic})
\]

where \(\text{id}\) is the empty substitution, \(\emptyset\) is the empty context and \(r\) and \(s\) are proofs of reflexivity and transitivity of the inclusion \(\subset\) of contexts.

The resulting proof of completeness does not require any enumeration of formulas. In our computational meta-language, it produces normal proofs of \(\vdash A\) by replicating the structure of the initial proof of \(\models M A\), thus suitable for semantic normalisation.

The contents of the talk is covered by a work-in-progress paper.

References